# EP 219 Assignment 1

**Viraj Karambelkar**
150260003

**Nitin Srirang**
150260027

**Harikrishnan KP**
150260026

**Hrishikesh T**
150260023

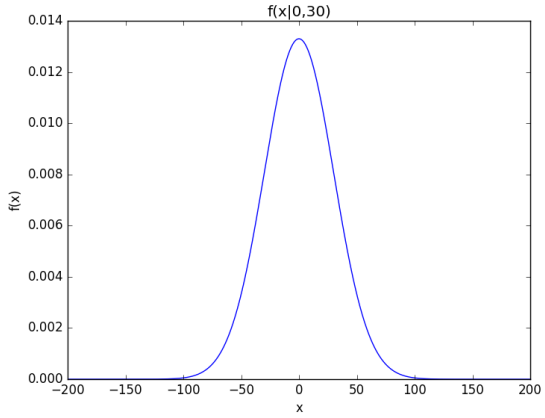
September 28, 2016

# Contents

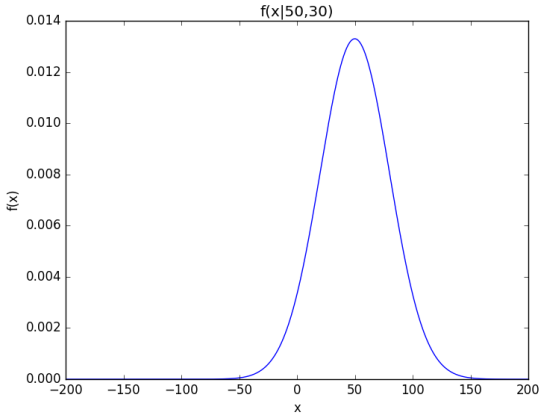
# 1 Report for Week 1:21st - 28th September 2016

## 1.1 Gaussian Plot

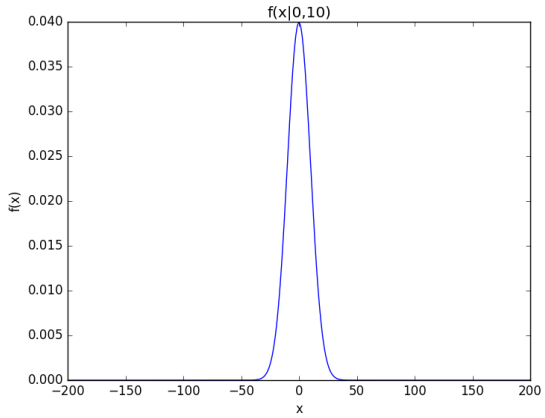
Gaussians with varying parameters were plotted using matplotlib's inbuilt Gaussian plot function. The parameters are the mean($\mu$) and the standard deviation($\sigma$).



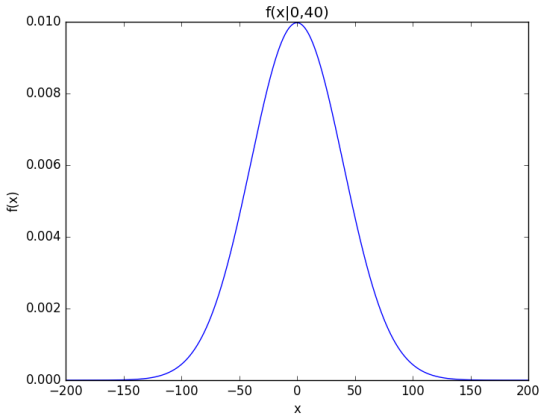

(a) $\mu = 0, \sigma = 30$




(b) $\mu = 50, \sigma = 30$




(c) $\mu = 0, \sigma = 10$




(d) $\mu = 0, \sigma = 40$

Figure 1: Plots of Gaussians with different means and variances

### 1.1.1 Description

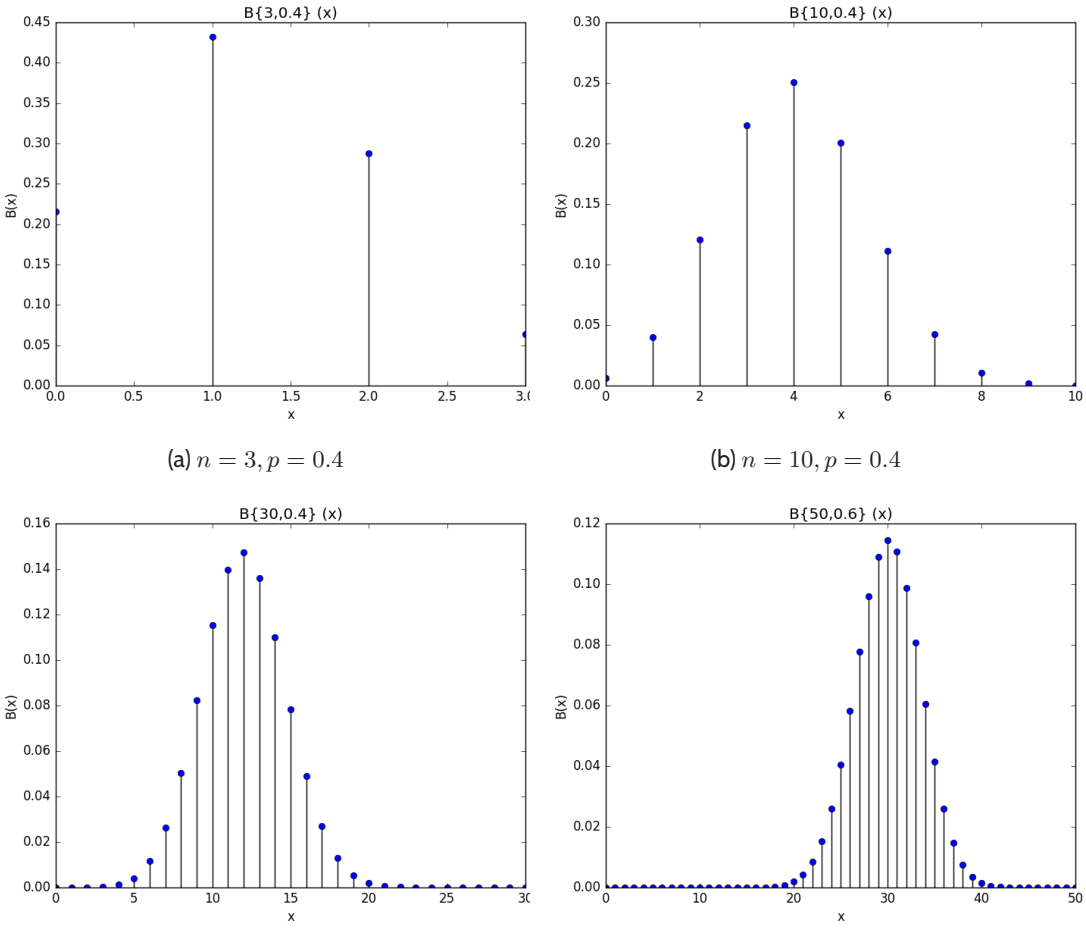
As is clear from 1a and 1b, the mean($\mu$) is a location parameter and the graph just gets shifted by 50units. Whereas, the standard deviation ($\sigma$) is a shape parameter. The width of the Gaussian increases/decreases as $\sigma$ increases/decreases respectively. This can be seen from 1a, 1c and 1d.

### 1.1.2 Code for Gaussian

```
# to calculate sqrt
import math
# to plot graph
import matplotlib.pyplot as plt
# to use the already defined normalized gaussian function
import matplotlib.mlab as mlab
# to define values of x to be plotted
import numpy as npn
mu = 0
variance = 100
sigma = math.sqrt(variance)
# 1000 values of x are plotted from x=-200 to x=200 )
x = np.linspace(-200,200,1000)

# predefined  normal function
plt.plot(x,mlab.normpdf(x,mu,sigma))
plt.xlabel('x')
plt.ylabel('f(x)')
# f(x| mean, standard deviation)
plt.title('f(x|0,5)')
plt.show()
```

## 1.2 Binomial Distribution

Binomials with varying n and p parameters i.e number of trials and probability of success in each trial have been plotted.




(a) $n = 3, p = 0.4$

(b) $n = 10, p = 0.4$

(c) $n = 30, p = 0.4$

(d) $n = 50, p = 0.4$

Figure 2: Binomial Distribution with varying n's and p's

### 1.2.1 Description

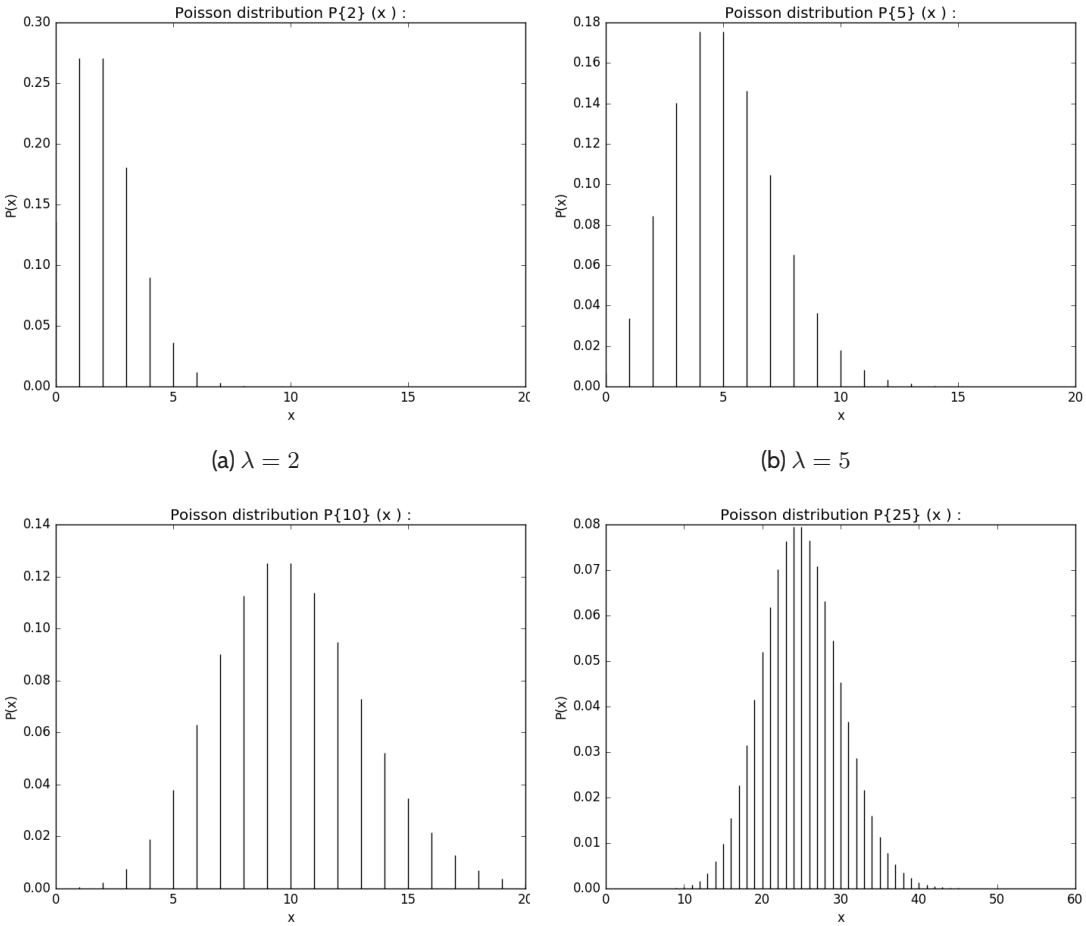
From Figure 2 it can be seen that the value of the mean i.e $np$ $(0.4, 4, 24, 30$ for Fig. 2a, Fig. 2b, Fig. 2c and Fig. 2d respectively) gives a very good estimate of the location of the maximum. Thus it behaves as a location parameter. The $\sigma$ i.e $np(1-p)$ $(0.48, 2.4, 4.8, 12$ for Fig. 2a, Fig. 2b, Fig. 2c and Fig. 2d respectively) give a very good indication of the width especially with increased number of trials. Also, the convergence of the Binomial to a Gaussian is also clearly seen in case of Fig ??.

### 1.2.2 Code for Binomial

```
import math
import numpy as np
import matplotlib.pyplot as plt

#defining a function to calculate factorial
def f(i):
j=1
k=1
while(j<=i):
k=k*j
j=j+1
return k
# defining a function to calculate nCr
def nCr(n, r):
return f(n)/(f(r)*f(n-r))
# n is obtained
n = int(input('enter the number of trials : '))
# p is entered
p = float(input('enter probability of success: '))

# x is an integer from 0 to n including both - a total of n+1 values
x = np.linspace(0,n,n+1)
y = []

# assigning B{n,p}(x) of each x to y
i=0
while (i<n+1):
y.append(nCr(n,i)*(p**i)*((1-p)**(n-i)))
i += 1

plt.plot(x,y,'bo')
plt.vlines(x,[0],y)
plt.title('B{'+str(n)','+str(p)+'} (x)')
plt.xlabel('x')
plt.ylabel('B(x)')
plt.show()
```

## 1.3 Poisson Distribution

The parameter involved is the mean $\lambda$. Probability that k events happen in a time t when $\lambda$ events occur in same t on an average.




(a) $\lambda = 2$ (b) $\lambda = 5$

(c) $\lambda = 10$ (d) $\lambda = 25$

Figure 3: Poisson Distribution

### 1.3.1 Description

Here, $\lambda$ is the sole parameter. The mean is given by $\lambda$ and $\sigma$ by $\sqrt{\lambda}$. For greater values of $\lambda$ as seen in Fig. 3d, the distribution looks more and more like a Gaussian of $\mu = 25$ and $\sigma = 5$.

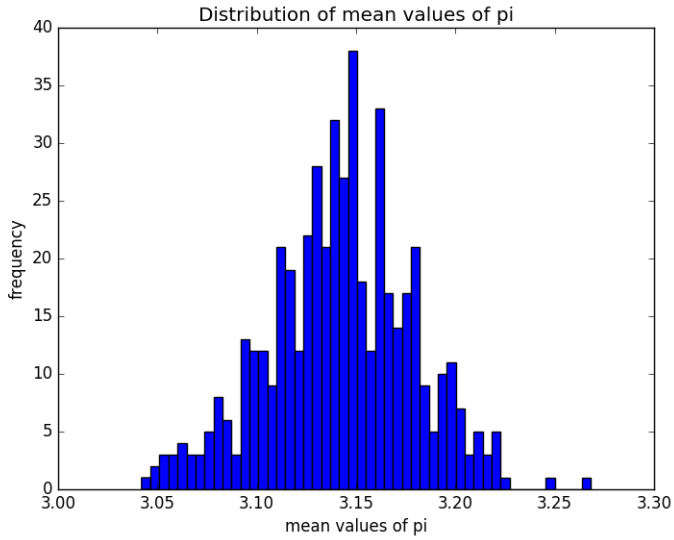
### 1.3.2 Code for Poisson

```
# to use the mathematical constant e
import math

# to assign values to the abscissa of the plot
import numpy as np

#to plot the distribution
import matplotlib.pyplot as plt

# lambda = average number of events in a fixed time
l = int(input('enter average no. of events in a fixed time interval: '))
y = []

# iterated k+1 times to include integers in the interval [0,k]
i=0
while (i<21):
# P(x) = e^(-l).(l^x)/x!
a = (math.exp(-l)*(l**i))/math.factorial(i)
y.append(a)
i += 1

plt.vlines(x,[0],y)
plt.title('Poisson distribution P{'+str(l)+'} (x ) :')
plt.xlabel('x')
plt.ylabel('P(x)')
plt.show()
```

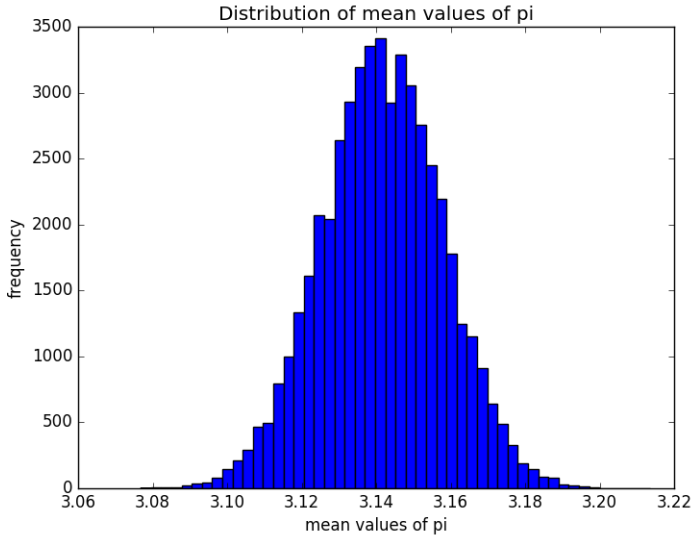
## 1.4 Distribution of $pis$

- Consider a square with center $(1, 1)$ of side 2.
- N($= 2000$) pairs of random numbers were generated using python's random number generator-Python generates numbers from $(0, 1)$ by $2$ so that the range of numbers goes from $(0, 1)$ to $(0, 2)$.
- The function "numincirc" now evaluates the distance of the generated ordered pair from the point(1,1) i.e the centre of the supposed square. If this distance is less than 1, it returns the boolean "True", else "False".
- The number of points returning True are recorded by the variable circnum.
- The value of $pi$ is proportional to the area of the circle by the area of the square which in turn is equal to the number of points inside the circle by the number of points inside the square.
- Hence, pi(observed)/4 = circnum/N.
- These values are now plotted as a histogram over 500 experiments.
- The mean obtained is $3.1419960000000002$.
- The measured error i.e sample width is $0.018152871932002674$
- The error i.e standard deviation obtained is $0.01836016821$.

Update: The number of experiments we perform doesn't affect the error in the measurement of pi. Theoretical error for both plots is $\sqrt{\frac{p\times(1-p)}{number\quad of\quad random\quad numbers}}$ $(= 0.01836016821)$. Error in the *mean* value of the pi obtained(w.r.t the actual value of pi ) is inversely proportional to $\sqrt{number\quad of\quad experiments}$. That is why comparing Fig. 4a and Fig. 4b we see a similar sample width. The corresponding plots are shown in the next page.




(a) N=2000, Num of Experiments = 500, Error = 0.018152871932002674




(b) N=10000, Num of Experiments = 50000,
Mean = 3.1415548880000146, Error in *mean* value of pi = 0.00410545841934081
Error(Theoretical) in value of pi = 0.01836016821

### 1.4.1 Description

With increase in the number of experiments, we can see a greater convergence to a Gaussian. This is as per the central limit theorem. This is observed very clearly from the difference between Fig. 4a and Fig. 4b.

### 1.4.2 Code for Monte Carlo

```
# for square root and actual pi value
import math
# for generating random numbers(points)
import random
# for making plots
import matplotlib.pyplot as plt
#function evaluates distance of point from (1,1) and reutrn True or false
def numincirc(x, y):
if (math.sqrt( (x-1)**2 + (y-1)**2)<= 1):
return True
else:
return False

 # creating and writing a file to record the 500 obtained values of meanpi
f = open('meanpifile9.txt','a')
#pi obtained from each run is stored
meanpi = []
# 500 experiments
num_of_exp=50000
N=10000
j=0
while (j<num_of_exp):

i=0
# counter to count the number of points inside Incircle
circnum = 0
while (i<N):
# assigning a random decimal numbers between 0 and 2 to x coordinate of point
x = random.random()*2
# similarly, for the y coord.
y = random.random()*2
point = numincirc(x, y)
if (point==True):
circnum += 1
i += 1

#probability(point in circle = Area(Circle)/Area(Square)) = pi/4.
#=> circnum/N = pi/4 => pi = 4*(circnum/N)
pi = 4*(circnum/float(N))
meanpi.append(pi)
# writing the value of pi to the text file
f.write('%f' % pi)
```

```
f.write(' ')
j += 1

meansum = 0.0
for x in meanpi:
meansum = meansum + x
mean = meansum/num_of_exp
print ("Mean is "+str(mean))
# Find the sample width of the points
mperror=0.0
for x in meanpi:
mperror=mperror + (x-mean)*(x-mean)
# mperror => mean pi error =standard deviation.
mperror = math.sqrt(mperror/(N-1))
# theoretical error in measurement = sigma(meanpi) = sqrt [(p*(1-p)/N]
#where p= pi(actual)/4
#a = p*(1-p)
a=(math.pi/4)*(1-(math.pi/4))
tperror = math.sqrt(4*a/N)
print ('Theoretical error is ' + str(tperror))
print ('The error in measurement is : '+str(mperror))

f.close()

# histogram with : mean pi values, no.of bins, type and width of each bar - is plotted
plt.hist(meanpi, 50, histtype='bar', rwidth=1)

plt.xlabel('mean values of pi')
plt.ylabel('frequency')
plt.legend()
plt.title('Distribution of mean values of pi')
plt.show()
```

## 2 Summary

1. Distributions:
    - Gaussian, Binomial and Poisson Distributions were plotted with varying parameters. It can clearly be seen from Fig. 1, Fig. 2, Fig. 3 that the mean $\mu$ is a location parameter and the standard deviation $\sigma$ is a width parameter(shape).
    - In case of the Binomial distributions, with an increase in $n$ we can see it converging to a Gaussian.
    - Similarly, in the case of the Poisson distribution, with an increase in $\lambda$ we can see th distribution converging to a Gaussian.
2. Distribution of mean values of pi:
    - Using the Monte Carlo method, 2000 random numbers drawn from a uniform distribution were placed inside a square of side 2 with center (1,1) in the x-y plane.
    - For each ordered pair (x,y), the distance from the point (1,1) was computed and compared to the number 1.
    - The number of such points lying inside the circle of radius 1 about (1,1) were found.
    - Now, the Area of Circle/Area of square = $\pi/4$. And ratio of areas = ratio of number of points in each(as these numbers are drawn from a uniform distribution).
    - This was repeated 500 times.
    - Thus the value of $\pi$ was estimated and a histogram was made for the same.
    - We observed Mean = $3.1419960000000002$ and Sample Width = $0.018152871932002674$

## 3 Team Responsibilities

The responsibilities of the team members for the current week (21st-28th September) are as follows:

| Team Sheet | |
|---|---|
| Viraj Karambelkar[1] | Group Leader |
| Nitin Srirang[2] | Programmer |
| Harikrishnan KP[3] | Web Manager |
| Hrishikesh Iyer[4] | Report Writer |

## 4 Website

The link to our website is The WIMPy Kids - Bringing Numbers to Life . All the assignments and the summary will be uploaded to the same. Each assignment will be uploaded to the Assignments Section and the weekly reports to the Report Section

## References


[1] Viraj Karambelkar, 150260003, Gmail

[2] Nitin Srirang, 150260027, Gmail

[3] Hari KP, 150260026, Gmail

[4] Hrishikesh T Iyer, 150260023, Gmail